

Neuroimaging Analysis Using Deep Learning for Brain Tumor Detection

Presented by, Under the Guidance of

LAVANYA PATIL, Dr. SAVITA S G

Dept. Of Computer Science and engineering (MCA), (Assistant Professor)

VTU CPGS Kalaburagi.

Lavanyapatil5109@gmail.com

Abstract— The motivation behind this study is to detect brain tumour and provide better treatment for the sufferings. The abnormal growths of cells in the brain are called tumours and cancer is a term used to represent malignant tumours. Usually CT or MRI scans are used for the detection of cancer regions in the brain. Positron Emission Tomography, Cerebral Arteriogram, Lumbar Puncture, Molecular testing are also used for brain tumour detection. In this study, MRI scan images are taken to analyse the disease condition. Objective this research works are i) identify the abnormal image ii) segment tumour region. Density of the tumour can be estimated from the segmented mask and it will help in therapy. Deep learning technique is employed to detect abnormality from MRI images. Multi level thresholding is applied to segment the

tumour region. Number of malignant pixels gives the density of the affected region.

Keywords: CT or MRI scans, malignant tumours

INTRODUCTION

Deep Learning:

Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction. These methods have dramatically improved the state-of-the-art in speech recognition, visual object recognition, object detection and many other domains such as drug discovery and genomics. Deep learning discovers intricate structure in large data sets by using the back propagation algorithm to indicate how a machine should change its internal parameters that are used to compute the representation in each layer from the representation in the previous layer. Deep convolutional nets have

brought about breakthroughs in processing images, video, speech and audio, whereas recurrent nets have shone light on sequential data such as text and speech.

Machine-learning technology powers many aspects of modern society: from web searches to content filtering on social networks to recommendations on e-commerce websites, and it is increasingly present in consumer products such as cameras and smartphones. Machine-learning systems are used to identify objects in images, transcribe speech into text, match news items, posts or products with users' interests, and select relevant results of search. Increasingly, these applications make use of a class of techniques called deep learning. Conventional machine-learning techniques were limited in their ability to process natural data in their raw form. For decades, constructing a pattern-recognition or machine-learning system required careful engineering and considerable domain expertise to design a feature extractor that transformed the raw data (such as the pixel values of an image) into a suitable internal representation or feature vector from which the learning subsystem, often a classifier, could detect or classify patterns in the input. Representation learning is a set of methods that allows a machine to be fed with raw

data and to automatically discover the representations needed for detection or classification. Deep-learning methods are representation-learning methods with multiple levels of representation, obtained by composing simple but non-linear modules that each transform the representation at one level (starting with the raw input) into a representation at a higher, slightly more abstract level. With the composition of enough such transformations, very complex functions can be learned. For classification tasks, higher layers of representation amplify aspects of the input that are important for discrimination and suppress irrelevant variations. An image, for example, comes in the form of an array of pixel values, and the learned features in the first layer of representation typically represent the presence or absence of edges at particular orientations and locations in the image. The second layer typically detects motifs by spotting particular arrangements of edges, regardless of small variations in the edge positions. The third layer may assemble motifs into larger combinations that correspond to parts of familiar objects, and subsequent layers would detect objects as combinations of these parts. The key aspect of deep learning is that these layers of features are not designed by human engineers: they are learned from data using

a general-purpose learning procedure. Deep learning is making major advances in solving problems that have resisted the best attempts of the artificial intelligence community for many years. It has turned out to be very good at discovering intricate structures in high-dimensional data and is therefore applicable to many domains of science, business and government. In addition to beating records in image recognition and speech recognition, it has beaten other machine-learning techniques at predicting the activity of potential drug molecules, analysing particle accelerator data, reconstructing brain circuits, and predicting the effects of mutations in non-coding DNA on gene expression and disease. Perhaps more surprisingly, deep learning has produced extremely promising results for various tasks in natural language understanding, particularly topic classification, sentiment analysis, question answering and language translation. We think that deep learning will have many more successes in the near future because it requires very little engineering by hand, so it can easily take advantage of increases in the amount of available computation and data. New learning algorithms and architectures that are currently being developed for deep neural networks will only accelerate this progress.

Supervised learning

The most common form of machine learning, deep or not, is supervised learning. Imagine that we want to build a system that can classify images as containing, say, a house, a car, a person or a pet. We first collect a large data set of images of houses, cars, people and pets, each labelled with its category. During training, the machine is shown an image and produces an output in the form of a vector of scores, one for each category. We want the desired category to have the highest score of all categories, but this is unlikely to happen before training. We compute an objective function that measures the error (or distance) between the output scores and the desired pattern of scores. The machine then modifies its internal adjustable parameters to reduce this error. These adjustable parameters, often called weights, are real numbers that can be seen as ‘knobs’ that define the input–output function of the machine. In a typical deep-learning system, there may be hundreds of millions of these adjustable weights, and hundreds of millions of labelled examples with which to train the machine. To properly adjust the weight vector, the learning algorithm computes a gradient vector that, for each weight, indicates by what amount the error would increase or decrease if the weight were increased by a

tiny amount. The weight vector is then adjusted in the opposite direction to the gradient vector. The objective function, averaged over all the training examples, can be seen as a kind of hilly landscape in the high-dimensional space of weight values. The negative gradient vector indicates the direction of steepest descent in this landscape, taking it closer to a minimum, where the output error is low on average. In practice, most practitioners use a procedure called stochastic gradient descent (SGD). This consists of showing the input vector for a few examples, computing the outputs and the errors, computing the average gradient for those examples, and adjusting the weights accordingly. The process is repeated for many small sets of examples from the training set until the average of the objective function stops decreasing. It is called stochastic because each small set of examples gives a noisy estimate of the average gradient over all examples. This simple procedure usually finds a good set of weights surprisingly quickly when compared with far more elaborate optimization techniques. After training, the performance of the system is measured on a different set of examples called a test set. This serves to test the generalization ability of the machine — its ability to produce sensible answers on new inputs that it has never seen during training.

Machine Learning

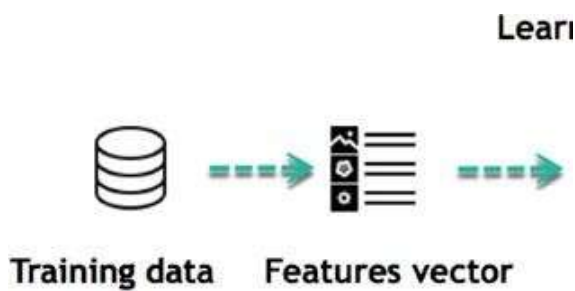
How does Machine Learning Work?

Machine learning is the brain where all the learning takes place. The way the machine learns is similar to the human being.

Humans learn from experience. The more we know, the more easily we can predict. By analogy, when we face an unknown situation, the likelihood of success is lower than the known situation. Machines are trained the same. To make an accurate prediction, the machine sees an example. When we give the machine a similar example, it can figure out the outcome. However, like a human, if its feed a previously unseen example, the machine has difficulties to predict.

The core objective of machine learning is the **learning** and **inference**. First of all, the machine learns through the discovery of patterns. This discovery is made thanks to the **data**. One crucial part of the data scientist is to choose carefully which data to provide to the machine. The list of attributes used to solve a problem is called a **feature vector**. You can think of a feature vector as a subset of data that is used to tackle a problem.

The machine uses some fancy algorithms to simplify the reality and transform this discovery into a **model**. Therefore, the learning stage is used to describe the data and summarize it into a model.



For instance, the machine is trying to understand the relationship between the wage of an individual and the likelihood to go to a fancy restaurant. It turns out the machine finds a positive relationship between wage and going to a high-end restaurant: This is the model

Inferring

When the model is built, it is possible to test how powerful it is on never-seen-before data. The new data are transformed into a features vector, go through the model and give a prediction. This is all the beautiful part of machine learning. There is no need to update the rules or train again the model. You can use the model previously trained to make inference on new data.

Literature Survey

The 2016 World Health Organization Classification of Tumors of the Central Nervous System is both a conceptual and practical advance over its 2007 predecessor. For the first time, the WHO classification of CNS tumors uses molecular parameters in addition to

histology to define many tumor entities, thus formulating a concept for how CNS tumor diagnoses should be structured in the molecular era. As such, the 2016 CNS WHO presents major restructuring of the diffuse gliomas, medulloblastomas and other embryonal tumors, and incorporates new entities that are defined by both histology and molecular features, including glioblastoma, IDH-wildtype and glioblastoma, IDH-mutant; diffuse midline glioma, H3 K27M-mutant; RELA fusion-positive ependymoma; medulloblastoma, WNT-activated and medulloblastoma, SHH-activated; and embryonal tumour with multilayered rosettes, C19MC-altered. The 2016 edition has added newly recognized neoplasms, and has deleted some entities, variants and patterns that no longer have diagnostic and/or biological relevance. Other notable changes include the addition of brain invasion as a criterion for atypical meningioma and the introduction of a soft tissue-type grading system for the now combined entity of solitary fibrous tumour/hemangiopericytoma-a departure from the manner by which other CNS tumors are graded. Overall, it is hoped that the 2016 CNS WHO will facilitate clinical, experimental and epidemiological studies that will lead to improvements in the lives of patients with brain tumors.

Background: The time between experiencing symptoms and treatment in cancer diseases is a time of insecurity and despair. Brain tumour disease is a severe disease with dramatic manifestations and it is important that this time be kept as short as possible. **Methods:** A consecutive sample of 28 patients with malignant gliomas and their spouses were interviewed about symptom development, help-seeking and experiences of medical care. The cumulative development of their symptoms was described and factors acting as obstacles to medical care were identified. **Results:** Most spouses witnessed months of global dysfunction preceding the symptom leading to physician consultation. The patient factors 'less alien symptoms', 'personality change' and 'avoidance'; the spouse factors 'spouse's passivity' and 'spouse's successive adaptation'; and the physician factors 'reasonable alternative diagnosis', 'physician's inflexibility' and 'physician's personal values' were identified as obstacles on the pathway to appropriate medical care. The importance of acknowledging the power of the spouse as a provider of substantial information from everyday life facilitating differential diagnosis is stressed. The term "brain tumours" refers to a mixed group of neoplasms originating from intracranial tissues and the meninges with degrees of

malignancy ranging from benign to aggressive. Each type of tumour has its own biology, treatment, and prognosis and each is likely to be caused by different risk factors. Even "benign" tumours can be lethal due to their site in the brain, their ability to infiltrate locally, and their propensity to transform to malignancy. This makes the classification of brain tumours a difficult science and creates problems in describing the epidemiology of these conditions. Public perception generally fails to distinguish between different tumour subtypes and although treatments and prognosis may vary, the functional neurological consequences are frequently similar. This article will give an overview of the burden of brain tumours in the population, looking at the major subtypes where possible, in addition to giving a summary of current views on possible cause. Measurement of Health Related Quality of Life (HRQL) in brain tumour patients is important because brain tumours and brain tumour treatment usually affect physical, cognitive as well as emotional functioning. Measurement of HRQL is important for the understanding of disease burden and for the impact of specific tumour treatment. Quality of Life is a multidimensional concept consisting of physical, psychological and social phenomena. A large number of Quality of

Life instruments have been developed. The European Organization for Research and Treatment of Cancer Quality of Life Questionnaire (EORTC QLQ-C30) and the MOS Short-Form Health Survey are two frequently used general HRQL instruments. A specific brain tumour scale is the Brain Cancer Module, which is designed to be used in combination with general questionnaires. HRQL measurement and neuropsychological examination were used to investigate the impact of radiotherapy and surgery in low-grade glioma patients and the influence of tumour volume, tumour localization, performance status and age in both low-grade and high-grade glioma patients. With the impetuous advancement of informatics, human knowledge is unable to bridge the boundaries and human computer interaction is paving the way for new eras. Here, a real-time human gesture recognition using an automated technology called Computer Vision is demonstrated. This is a type of noncognitive computer user interface, having the endowment to perceive gestures and execute commands based on that. The design is implemented on a Linux system but can be implemented by installing modules for python on a windows system also. OpenCV and KERAS are the platforms used for the identification. Gesture displayed in the

screen is recognized by the vision-based algorithms. Using background removal technique, an assortment of skin color masks was trained by Lenet architecture in KERAS for the recognition. The users have tested and produced over 5000 masks with KERAS to generate 96% more accurate results.

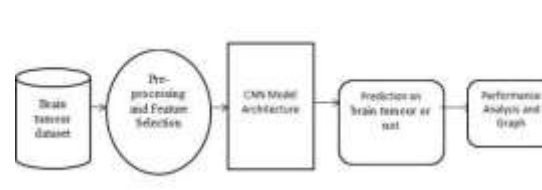


Fig 1. Proposed architecture

EXISTING WORK

Mircea Gurbin, Mihaela Lascu, and Dan Lascu et al. proposed a method consisting of Continuous Wavelet Transform (CWT), Discrete Wavelet Transform (DWT) and Support Vector Machine (SVM). It uses different levels of wavelets, and by training, the cancerous and non-cancerous tumours can be identified. The computation time is longer for the proposed method.

Somasundaram S. and Gobinath R. et al. explains the present status of detection and segmentation of tumour through deep learning models. For deeper segmentation, 3D based CNN, ANN and SVM is used.

Damodharan S. and Raghavan D. et al. address segmentation of pathological tissues (Tumor), normal tissues (White Matter (WM) and Gray Matter (GM)) and fluid (Cerebrospinal Fluid (CSF)), extraction of the relevant features from each segmented tissues and classification of the tumor images with Neural Network (NN).

PROPOSED METHODOLOGY

The aim of this system is to build a system that would help in brain tumour detection from MRI images through the convolution neural network. The proposed method was tested and compared with the existing classification techniques to determine the accuracy of the proposed method.

Deep learning is a machine learning technique that instructs computers what to do as a human think and do in a scenario. In deep learning, a computer model is able to do classification tasks from images, sound or text. Sometimes human level performance is being exceeded by deep learning techniques. One of the most popular neural networks is an artificial neural network that has a collection of simulated neurons. Each neuron acts as a

node and by links each node is connected to other nodes.

The proposed system consists of image acquisition, preprocessing, segmentation, feature extraction and classification.

This algorithm is faster in execution for normal MRI images.

MODULES DESCRIPTION:

1Modules

- ❖ Dataset
- ❖ Importing the necessary libraries
- ❖ Retrieving the images
- ❖ Splitting the dataset
- ❖ Building the model
- ❖ Apply the model and plot the graphs for accuracy and loss
- ❖ Accuracy on test set
- ❖ Saving the Trained Model

MODULES DESCRIPTION:

Dataset:

In the first module, we developed the system to get the input dataset for the training and testing purpose. We have taken the dataset from brain tumour detection Link

<https://www.kaggle.com/navoneel/brain-mri-images-for-brain-tumor-detection>

The dataset consists of 3000 brain tumour images

Importing the necessary libraries:

We will be using Python language for this. First we will import the necessary libraries such as keras for building the main model, sklearn for splitting the training and test data, PIL for converting the images into array of numbers and other libraries such as pandas, numpy ,matplotlib and tensorflow.

Retrieving the images:

We will retrieve the images and their labels. Then resize the images to (224,224) as all images should have same size for recognition. Then convert the images into numpy array.

Splitting the dataset:

Split the dataset into train and test. 80% train data and 20% test data.

Convolutional Neural Networks

The objectives behind the first module of the course 4 are:

- To understand the convolution operation
- To understand the pooling operation
- Remembering the vocabulary used in convolutional neural networks (padding, stride, filter, etc.)

- Building a convolutional neural network for multi-class classification in images

Computer Vision

Some of the computer vision problems which we will be solving in this article are:

1. Image classification
2. Object detection
3. Neural style transfer

One major problem with computer vision problems is that the input data can get really big. Suppose an image is of the size 68 X 68 X 3. The input feature dimension then becomes 12,288. This will be even bigger if we have larger images (say, of size 720 X 720 X 3). Now, if we pass such a big input to a neural network, the number of parameters will swell up to a HUGE number (depending on the number of hidden layers and hidden units). This will result in more computational and memory requirements – not something most of us can deal with.

Edge Detection Example

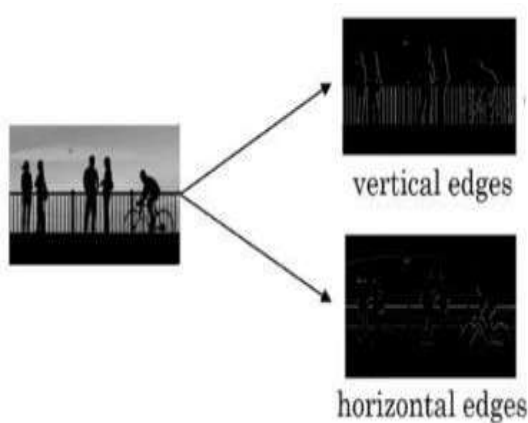
In the previous article, we saw that the early layers of a neural network detect edges from an image. Deeper layers might be able to detect the cause of the objects and even more deeper layers might detect the cause of complete objects (like a person's face).

In this section, we will focus on how the edges can be detected from an image.

Suppose we are given the below image:



As you can see, there are many vertical and horizontal edges in the image. The first thing to do is to detect these edges:



But how do we detect these edges? To illustrate this, let's take a 6 X 6 grayscale image (i.e. only one channel):

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

Next, we convolve this 6 X 6 matrix with a 3 X 3 filter:

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

6 X 6 image

1	0	-1
1	0	-1
1	0	-1

3 X 3 filter

After the convolution, we will get a 4 X 4 image. The first element of the 4 X 4 matrix will be calculated as:

3 ¹	0 ⁰	1 ⁻¹
1 ¹	5 ⁰	8 ⁻¹
2 ¹	7 ⁰	2 ⁻¹

So, we take the first 3 X 3 matrix from the 6 X 6 image and multiply it with the filter. Now, the first element of the 4 X 4 output will be the sum of the element-wise product of these values, i.e. $3*1 + 0 + 1*-1 + 1*1 + 5*0 + 8*-1 + 2*1 + 7*0 + 2*-1 = -5$. To calculate the second element of the 4 X 4 output, we will shift our filter one step towards the right and again get the sum of the element-wise product:

0 ¹	1 ⁰	2 ⁻¹
5 ¹	8 ⁰	9 ⁻¹
7 ¹	2 ⁰	5 ⁻¹

Similarly, we will convolve over the entire image and get a 4 X 4 output:

-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

So, convolving a 6 X 6 input with a 3 X 3 filter gave us an output of 4 X 4. Consider one more example:

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

1	0	-1
1	0	-1
1	0	-1

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

1	0	-1
1	0	-1
1	0	-1

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

Note: Higher pixel values represent the brighter portion of the image and the lower pixel values represent the darker portions. This is how we can detect a vertical edge in an image.

More Edge Detection

The type of filter that we choose helps to detect the vertical or horizontal edges. We can use the following filters to detect different edges:

1	0	-1
1	0	-1
1	0	-1

Vertical

1	1	1
0	0	0
-1	-1	-1

Horizontal

Some of the commonly used filters are:

1	0	-1
2	0	-2
1	0	-1

Sobel filter

3	0	-3
10	0	-10
3	0	-3

Scharr filter

The Sobel filter puts a little bit more weight on the central pixels. Instead of using these filters, we can create our own as well and treat them as a parameter which the model will learn using backpropagation.

Padding

We have seen that convolving an input of 6 X 6 dimension with a 3 X 3 filter results in 4 X 4 output. We can generalize it and say that if the input is $n \times n$ and the filter size is $f \times f$, then the output size will be $(n-f+1) \times (n-f+1)$:

- **Input:** $n \times n$
- **Filter size:** $f \times f$
- **Output:** $(n-f+1) \times (n-f+1)$

There are primarily two disadvantages here:

1. Every time we apply a convolutional operation, the size of the image shrinks

1. Pixels present in the corner of the image are used only a few numbers of times during convolution as compared to the central pixels. Hence, we do not focus too much on the corners since that can lead to information loss

To overcome these issues, we can pad the image with an additional border, i.e., we add one pixel all around the edges. This means that the input will be an 8×8 matrix (instead of a 6×6 matrix). Applying convolution of 3×3 on it will result in a 6×6 matrix which is the original shape of the image. This is where padding comes to the fore:

- **Input:** $n \times n$
- **Padding:** p
- **Filter size:** $f \times f$
- **Output:** $(n+2p-f+1) \times (n+2p-f+1)$

There are two common choices for padding:

1.Valid: It means no padding. If we are using valid padding, the output will be $(n-f+1) \times (n-f+1)$

1.Same: Here, we apply padding so that the output size is the same as the input size, i.e., $n+2p-f+1=n$

So, $p = (f-1)/2$

We now know how to use padded convolution. This way we don't lose a lot of information and the image does not shrink

either. Next, we will look at how to implement strided convolutions.

Strided Convolutions

Suppose we choose a stride of 2. So, while convoluting through the image, we will take two steps – both in the horizontal and vertical directions separately. The dimensions for stride s will be:

Input: $n \times n$

Padding: p **Stride:**

s

Filter size: $f \times f$

Output: $[(n+2p-f)/s+1] \times [(n+2p-f)/s+1]$

Stride helps to reduce the size of the image, a particularly useful feature.

Convolutions Over Volume

Suppose, instead of a 2-D image, we have a 3-D input image of shape $6 \times 6 \times 3$. How will we apply convolution on this image? We will use a $3 \times 3 \times 3$ filter instead of a 3×3 filter. Let's look at an example:

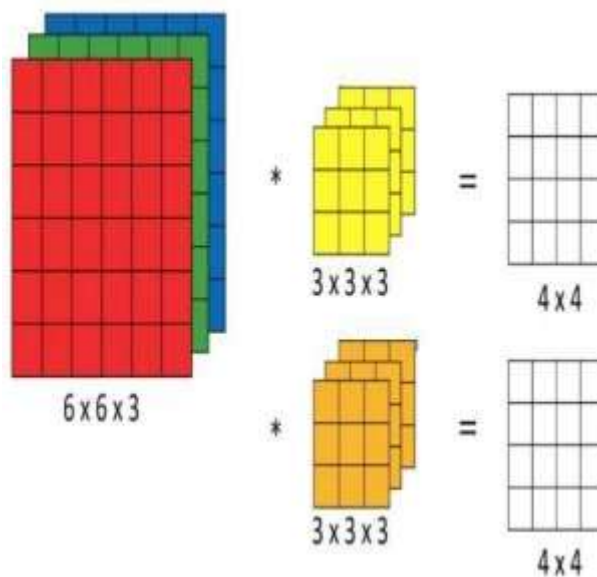
- **Input:** $6 \times 6 \times 3$
- **Filter:** $3 \times 3 \times 3$

The dimensions above represent the height, width and channels in the input and filter. Keep in mind that the number of channels in the input and filter should be same. This will result in an output of 4×4 . Let's understand it visually:

Since there are three channels in the input, the filter will consequently also have three

channels. After convolution, the output shape is a 4 X 4 matrix. So, the first element of the output is the sum of the element-wise product of the first 27 values from the input (9 values from each channel) and the 27 values from the filter. After that we convolve over the entire image.

Instead of using just a single filter, we can use multiple filters as well. How do we do that? Let's say the first filter will detect vertical edges and the second filter will detect horizontal edges from the image. If we use multiple filters, the output dimension will change. So, instead of having a 4 X 4 output as in the above example, we would have a 4 X 4 X 2 output (if we have used 2 filters):



Generalized dimensions can be given as:

- **Input:** $n \times n \times n_c$
- **Filter:** $f \times f \times n_c$
- **Padding:** p

- **Stride:** s
- **Output:** $[(n+2p-f)/s+1] \times [(n+2p-f)/s+1] \times n_c'$

Here, n_c is the number of channels in the input and filter, while n_c' is the number of filters.

One Layer of a Convolutional Network

Once we get an output after convolving over the entire image using a filter, we add a bias term to those outputs and finally apply an activation function to generate activations. *This is one layer of a convolutional network.* Recall that the equation for one forward pass is given by:

$$z^{[1]} = w^{[1]} * a^{[0]} + b^{[1]}$$

$$a^{[1]} = g(z^{[1]})$$

In our case, input (6 X 6 X 3) is $a^{[0]}$ and filters (3 X 3 X 3) are the weights $w^{[1]}$. These activations from layer 1 act as the input for layer 2, and so on. Clearly, the number of parameters in case of convolutional neural networks is independent of the size of the image. It essentially depends on the filter size. Suppose we have 10 filters, each of shape 3 X 3 X 3. What will be the number of parameters in that layer? Let's try to solve this:

- Number of parameters for each filter = $3*3*3 = 27$
- There will be a bias term for each filter, so total parameters per filter = 28

- As there are 10 filters, the total parameters for that layer = $28 \times 10 = 280$

No matter how big the image is, the parameters only depend on the filter size. Awesome, isn't it? Let's have a look at the summary of notations for a convolution layer:

- $f^{[l]}$ = filter size
- $p^{[l]}$ = padding
- $s^{[l]}$ = stride
- $n_{[c]}^{[l]}$ = number of filters

Let's combine all the concepts we have learned so far and look at a convolutional network example.

Simple Convolutional Network Example

We'll take things up a notch now. Let's look at how a convolution neural network with convolutional and pooling layer works. Suppose we have an input of shape $32 \times 32 \times 3$:

We take an input image (size = $39 \times 39 \times 3$ in our case), convolve it with 10 filters of size 3×3 , and take the stride as 1 and no padding. This will give us an output of $37 \times 37 \times 10$. We convolve this output further and get an output of $7 \times 7 \times 40$ as shown above. Finally, we take all these numbers ($7 \times 7 \times 40 = 1960$), unroll them into a large vector, and pass them to a classifier that will make predictions. This is a microcosm of how a convolutional network works.

There are a number of hyperparameters that we can tweak while building a convolutional network. These include the number of filters, size of filters, stride to be used, padding, etc. We will look at each of these in detail later in this article. Just keep in mind that as we go deeper into the network, the size of the image shrinks whereas the number of channels usually increases.

In a convolutional network (ConvNet), there are basically three types of layers:

1. Convolution layer
2. Pooling layer
3. Fully connected layer

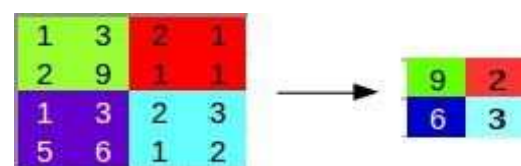
Let's understand the pooling layer in the next section.

Pooling Layers

Pooling layers are generally used to reduce the size of the inputs and hence speed up the computation. Consider a 4×4 matrix as shown below:

1	3	2	1
2	9	1	1
1	3	2	3
5	6	1	2

Applying max pooling on this matrix will result in a 2×2 output:



For every consecutive 2 X 2 block, we take the max number. Here, we have applied a filter of size 2 and a stride of 2. These are the hyperparameters for the pooling layer.

Results

Brain tumor detection using deep learning represents a significant advancement in medical imaging and diagnostic accuracy. Deep learning, particularly Convolutional Neural Networks (CNNs), is employed to analyze medical images such as MRI scans to identify and classify brain tumors with high precision. These models are trained on large datasets of labeled brain images, where tumors are annotated by medical professionals. The primary advantage of deep learning in this context is its ability to learn intricate patterns and features that may not be visible to the human eye or traditional image analysis methods. The process typically involves multiple steps: preprocessing the MRI images to enhance quality, segmenting the brain region, and then applying CNNs to detect and classify tumors into different categories, such as benign or malignant. Advanced models may incorporate techniques like transfer learning, where models pre-trained on large, general datasets are fine-tuned for the specific task of tumor detection, thus

reducing the need for extensive medical data. Moreover, some systems use 3D CNNs to analyze volumetric data from the MRI scans, providing a more comprehensive understanding of the tumor's size, shape, and location. This three-dimensional approach is particularly beneficial for assessing the tumor's growth and planning treatment strategies.

Deep learning models for brain tumor detection have demonstrated promising results, achieving high accuracy and sensitivity, which are critical for early diagnosis and treatment. However, challenges such as data scarcity, variability in imaging protocols, and the need for interpretability and trustworthiness of AI predictions still need to be addressed. Continued advancements in this field hold the potential to significantly improve patient outcomes and support healthcare professionals in making more informed and timely decisions.

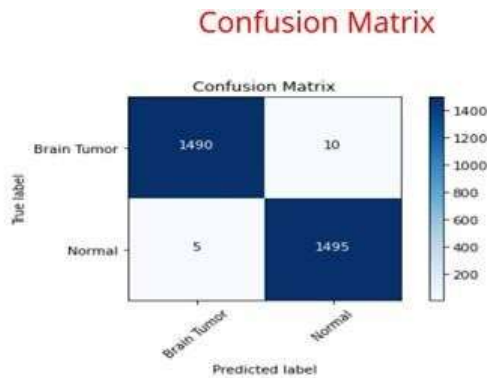


Fig:2 Result Matrix

CONCLUSION

This paper provides a new method for detecting brain tumour by deep learning method. The early detection of cancer helps timely and effective treatment. Kaggle dataset contains good quality of MRI images for research purposes. Different segmentation algorithms were experimented. From this, multilevel thresholding and OTSU thresholding are the best methods for the dataset. Convolutional Neural Network with modified approach helped to get a result with accuracy 98%.

Density estimation method is also proposed using Gaussian kernel distribution. This system can be improved to support with a web interface. Detection of different diseases can be also identified from the MRI images. Apart from the density some other parameters can also estimated for therapeutic purposes.

REFERENCES

- [1] David N. Louis, Arie Perry, et al. , “The 2016 World Health Organization Classification of Tumors of the Central Nervous System: a summary” , Acta Neuropathol , Springer may 2016
- [2] Pär Salander, A Tommy Bergenheim, Katarina Hamberg, Roger Henriksson, Pathways from symptoms to medical care: a descriptive study of symptom development and obstacles to early diagnosis in brain tumour patients, Family Practice, Volume 16, Issue 2, April 1999, Pages 143–148
- [3] McKinney PA ,”Brain tumours: incidence, survival, and aetiology”,Journal of Neurology, Neurosurgery & Psychiatry 2004;75:ii12-ii17.
- [4] Heimans, J., Taphoorn, M. Impact of brain tumour treatment on quality of life. J Neurol 249, 955–960 (2002)
- [5] Malavika Suresh, et al. “Real-Time Hand Gesture Recognition Using Deep Learning”, International Journal of Innovations and Implementations in Engineering(ISSN 2454- 3489), 2019, vol 1

[6] M. Gurbină, M. Lascu and D. Lascu,
“Tumor Detection and Classification of
MRI Brain Image using Different Wavelet
Transforms and Support Vector Machines”,
42nd International Conference on
Telecommunications and Signal Processing
(TSP), Budapest, Hungary, 2019